Fundamentos de Algoritmia

Grados en Ingeniería Informática

Convocatoria extraordinaria, 7 de septiembre de 2020. Grupos B, D y E

1. (3 puntos) Dado vector `int v[n]` de enteros ($n >= 0$), se dice que una secuencia de valores consecutivos de v es un *tramo par* cuando todos los valores son números pares. Debe especificarse, diseñarse e implementarse un algoritmo iterativo que, dado el vector `int v[n]` determine la longitud (es decir, el número de valores) del *tramo par* más largo en v (*Nota:* En caso de que v no contenga tramos pares, el resultado deberá ser 0). Debe determinarse, asimismo, justificadamente el orden de complejidad del algoritmo.

La implementación deberá ir acompañada de un programa de prueba, que lea desde la entrada estandar casos de prueba, los ejecute, e imprima por la salida estándar el resultado. Cada caso de prueba consistirá en dos líneas: (i) en la primera se especificar el número de elementos n del vector (a lo sumo habrá vectores de 1000 elementos); y (ii) en la segunda línea aparecerán en orden los distintos valores el vector. Como resultado deberá imprimirse la longitud pedida. La salida finalizará con una línea en la que aparezca -1.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	**Salida**
10 2 4 5 7 9 6 8 10 12 14 5 2 4 5 8 10 3 1 3 5 -1	5 2 0

2. (3.5 puntos) Un numero natural se dice *interesante* cuando no contiene ningún cero, y cuando cada dígito divide a la suma de todos los dígitos que lo preceden, y también divide a la suma de todos los dígitos que lo suceden. Debe diseñarse e implementarse un algoritmo recursivo que, dado un número natural n, devuelva `true` si n es *interesante*, y `false` en caso contrario. Debe determinarse, asimismo, justificadamente el orden de complejidad del algoritmo.

La implementación deberá ir acompañada de un programa de prueba, que lea desde la entrada estandar casos de prueba, los ejecute, e imprima por la salida estándar el resultado. La entrada comenzará con el número de casos de prueba. Cada caso consistirá en una línea con el número n a analizar. El programa escribirá `SI` si el número n es *interesante*, o `NO` en otro caso.

A continuación se muestra un ejemplo de entrada / salida:

Entrada	**Salida**
3 621348 621356 846213	SI NO SI

3. (3.5 puntos) La agencia matrimonial `Tortolitos` necesita desarrollar un sistema que recomiende posibles emparejamientos entre sus clientes. Para ello, la agencia ha determinado las afinidades entre dichos clientes. Dichas afinidades vienen dadas por una matriz de números naturales

A. El elemento A_{ij} de dicha matriz contiene la afinidad que el cliente i siente por el cliente j. Los emparejamientos propuestos (*emparejamientos factibles*) deben cumplir las siguientes restricciones:

- No es posible emparejar un cliente consigo mismo.
- Todos los clientes deben tener pareja, y dicha pareja debe ser única.
- Si el cliente i está emparejado con el cliente j, entonces ambos clientes deben ser *compatibles*, en el sentido de que $A_{ij} > 0$ (es decir, el cliente i debe tener cierto grado de afinidad con el cliente j) y $A_{ji} > 0$ (es decir, el cliente j debe tener cierto grado de afinidad con el cliente i).

Por otra parte: (i) la afinidad de una pareja $\{i, j\}$ es la suma de la afinidad que i siente por j, y la afinidad que j siente por i (es decir, $A_{ij} + A_{ji}$); y (ii) la afinidad de un emparejamiento es la suma de las afinidades de todas las parejas que forman parte de dicho emparejamiento.

Se pide diseñar e implementar un algoritmo "vuelta atrás" que, dada una matriz de afinidad A, determine el máximo de las afinidades de todos los emparejamientos factibles (*Nota:* En caso de que no haya emparejamientos factibles, dicho valor deberá ser 0).

La implementación deberá ir acompañada de un programa de prueba, que lea desde la entrada estandar casos de prueba, los ejecute, e imprima por la salida estándar el resultado. La entrada comenzará con el número de casos de prueba. Cada caso comenzará con una línea que indicará el número n de clientes (un número par mayor que 0 y menor o igual que 10). A continuación aparecerá la matriz de afinidades, listada por filas (la primera línea contendrá la primera fila de la matriz con la afinidad del primer cliente con los demás, la segunda línea las afinidades del segundo cliente, etc.). La salida consistirá en una línea con un único valor, el máximo pedido.

A continuación se muestra un ejemplo de entrada / salida:

Entrada	**Salida**
3 2 0 10 8 0 4 0 1 1 1 0 0 0 0 9 8 0 10 5 4 3 0 4 0 10 8 7 7 0 8 0 9 6 0 1 10 0 9 0	18 0 31

Explicación del ejemplo: En el primer caso, hay solo dos clientes compatibles lo que implica un único emparejamiento factible, que tiene una única pareja. La afinidad del emparejamiento será 18 (la afinidad de dicha pareja, que, a su vez, es la suma de la afinidad que siente el primer cliente por el segundo, y la que siente el segundo cliente por el primero). Por tanto, dicha afinidad será también la máxima afinidad pedida. En el segundo ejemplo el segundo cliente no es afin a nadie, por lo que no hay emparejamientos factibles. Por tanto, el resultado debe ser 0. En el último caso el segundo y cuarto cliente solo pueden emparejarse con el primero o el tercero lo que hace que haya dos emparejamientos: (i) el primero con el segundo y el tercero con el cuarto, lo que da una afinidad de 27 y (ii) el primero con el cuarto y el segundo con el tercero, lo que da una afinidad de 31. Por tanto, la máxima afinidad es 31.